**** ALGORITMO DI BELLMAN-FORD

Individua il percorso a costo minimo tra un nodo sorgente e un nodo destinazione.

ES. BELLMAN-FORD

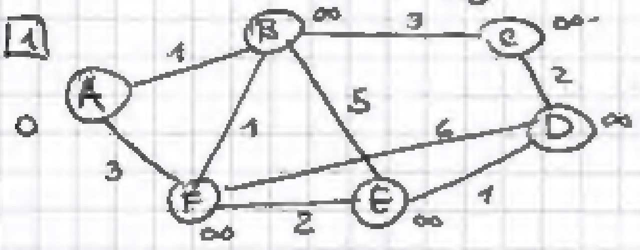
Assegno al ~~router~~ nodo radice costo ∅, e agli altri nodi costo ∞.
Procedo dal nodo radice etichettato con A, e cerco i suoi vicini, aggiorno i costi e scrivo il router predecessore a fianco di ogni router aggiornato.

Passo al nodo B e ripeto il procedimento, poi a C, D, ecc. fino a concludere il ciclo.

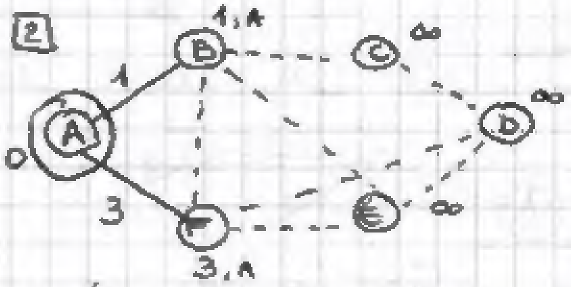
Reitero ripartendo da A

L'algoritmo si arresta al ciclo ennesimo senza alcun aggiornamento di costo.

1



2

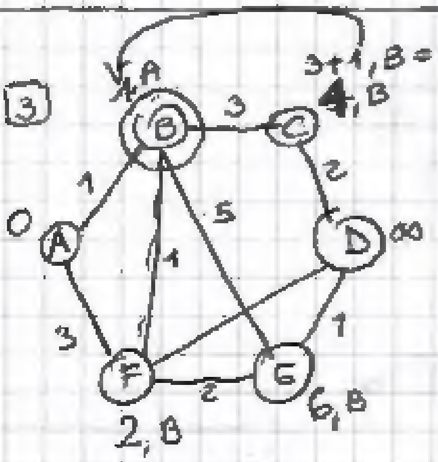


costo di B prima: ∞

dopo: costo di A + 1
= 0 + 1 = 1.

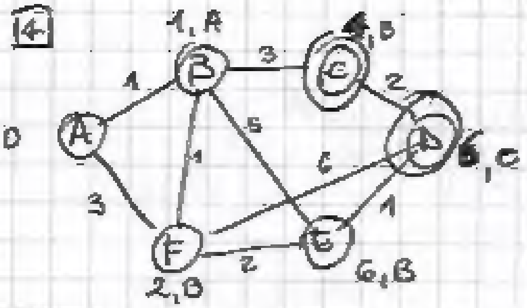
Aggiorno con A predecessore

3



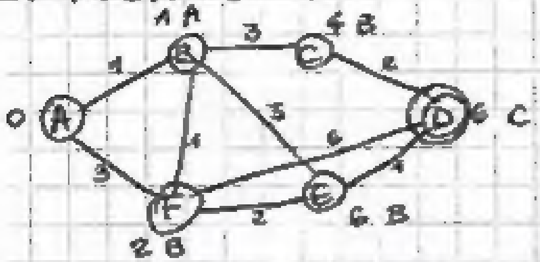
Prendo B e aggiorno i costi confrontando il costo precedente con la somma costo del nodo B + costo del collegamento. Riporto i nodi predecessori su C, E, F.

4

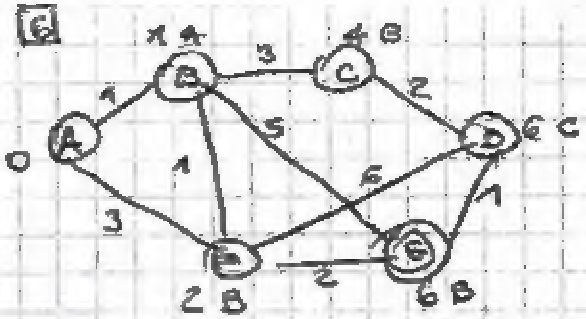


Prendo C

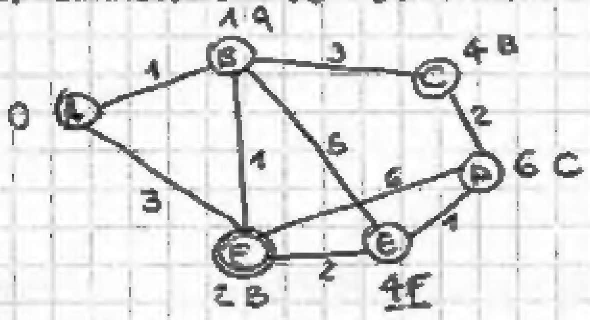
5 Prendo D



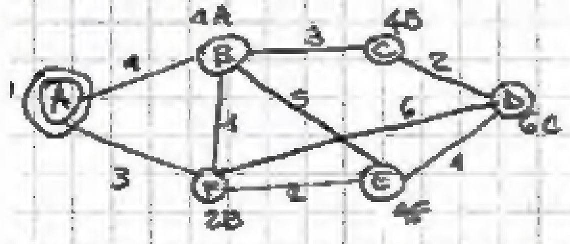
6



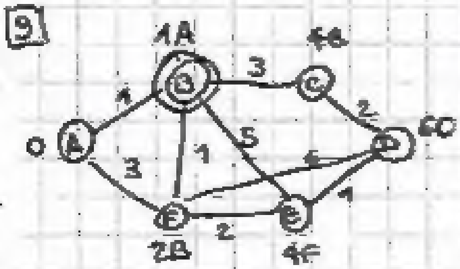
7 Chiudo il ciclo con il nodo F:



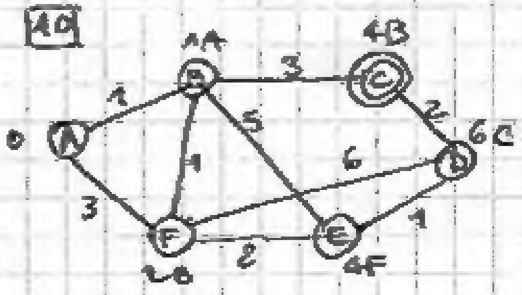
8 Inizio un nuovo ciclo da A



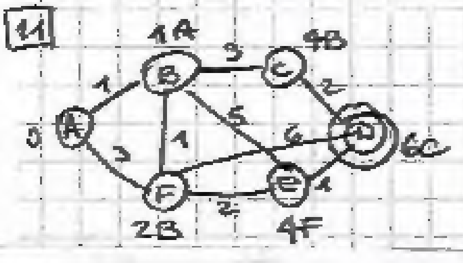
9



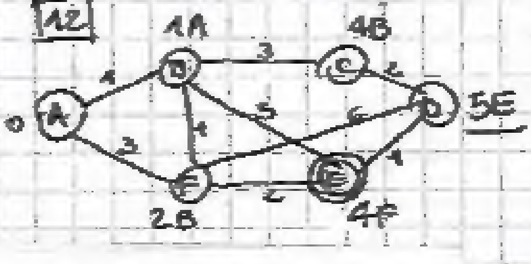
10



11

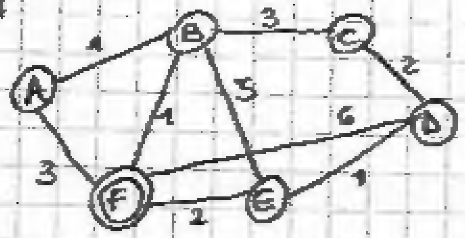


12



quindi sarà necessario (almeno) un altro ciclo

12



13 ÷ n

Al ciclo successivo non ci sono aggiornamenti. Quindi il grafo finale coi costi minimi è quello in 12.

***** ALGORITMO DI BELLMAN-FORD
****** VANTAGGI / SVANTAGGI

Se un router va KO:

BELLMAN-FORD: si aggiornano i costi della rete senza quel router, partendo dai costi trovati precedentemente.

DIJKSTRA: Bisogna ricalcolare tutto da capo

Dijkstra richiede una singola iterazione, mentre Bellman-Ford un numero n.

Dijkstra risente di più dei cambi di topologia, mentre Bellmann-Ford nelle sue iterazioni n-esime rileva con maggiore ~~probabilità~~ frequenza i cambi, e non richiede di ricalcolare i costi da zero per tutti i nodi.